IN THE CLAIMS:

1-66. (CANCELLED)

67. (PREVIOUSLY PRESENTED) A processor, comprising:

a first execution unit having a first and second input register coupled to first and second inputs to a first arithmetic logic unit (ALU), the first and second input registers of the first execution unit to store source operands, the first ALU capable of addressing a memory to retrieve a source operand;

a second execution unit having a first and second input register, the second register coupled to a second input to a second ALU, the first and second input registers of the second execution unit to store source operands, the second ALU not capable of addressing the memory; and

a multiplexer (MUX) having i) a first input coupled to the first input of the first ALU, ii) a second input coupled to the first input register of the second ALU, and iii) an output directly providing a first input to the second ALU, the MUX permitting both the first and second ALU to share the source operand retrieved by the first ALU from the memory.

68. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

an instruction set defining a register decode value that specifies source operand bypassing, such that the MUX, in response to the register decode value that specifies source operand bypassing, selects the first input of the MUX coupled to the first input of the first ALU as the output of the MUX, the output of the MUX providing the first input to the second ALU.

69. (PREVIOUSLY PRESENTED) The processor of claim 68, wherein the source operand bypassing value allows the second execution unit to receive data stored at an effective memory address specified by a displacement operand in the previous instruction executed by the first execution unit.

70. (CURRENTLY AMENDED) The processor of claim 67, further comprising:

a local bus for communicating with ~~a~~ the memory;

a register file for storing intermediate operands; and

an instruction decode stage for coupling the register file to the first and second input registers of the first and second ALUs to provide intermediate operands as the source operands, and for coupling a memory bus to the first input register of the of the first ALU to provide source operands from the memory.

71. (PREVIOUSLY PRESENTED) The processor of claim 70, wherein the first input register of the first ALU provides source operands from the memory to both the first input to the first ALU and to the first input of the MUX, thereby permitting the first input to the second ALU to share the source operands from the memory directly from the first input register of the first ALU.

72. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

a pipeline of the processor, the pipeline having a plurality of stages including instruction decode, writeback, and execution stages, the execution stage having the first and second execution units.

73. (PREVIOUSLY PRESENTED) The processor of claim 72, further comprising:

an instruction set defining a register decode value that defines result bypassing that allows bypassing of a result from a previous instruction executing in pipeline stages of the processor by directly addressing a result register of the first and second execution units.

74. (PREVIOUSLY PRESENTED) The processor of claim 73, wherein the register decode value comprises:

one of a result bypass (RRB) operand and an inter-unit result bypass (RIRB) operand, each of which explicitly controls data flow within the pipeline of the processor.

75. (PREVIOUSLY PRESENTED) The processor of claim 74, wherein the RRB operand denotes the first execution unit and the RIRB operand denotes the second execution unit.

76. (PREVIOUSLY PRESENTED) The processor of claim 74, wherein the RRB operand explicitly infers feedback of the data delivered from the first execution unit to an input register of the first execution unit over a feedback path.

77. (PREVIOUSLY PRESENTED) The processor of claim 76, wherein the writeback stage comprises an interstage register and wherein the RRB operand enables bypassing write-back of the data processed by the first and second execution units to one of the register file or the interstage register of the writeback stage.

78. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

an instruction set defining a register decode value that defines source operand bypassing that allows source operand data to be shared among the first and second execution units by directly addressing a source register of the first execution unit.

79. (PREVIOUSLY PRESENTED) The processor of claim 78, wherein the source operand bypassing value allows the second execution unit to receive data stored at an effective memory address specified by a displacement operand in the previous instruction executed by the first execution unit.

80. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

a register file containing a plurality of general-purpose registers for storing intermediate result data processed by the first and second execution units.

81. (PREVIOUSLY PRESENTED) The processor of claim 67, wherein the first and second execution units are parallel execution units.

82. (PREVIOUSLY PRESENTED) The processor of claim 67, further comprising:

a current execution unit as the first execution unit; and

an alternate execution unit as the second execution unit.

83. (PREVIOUSLY PRESENTED) The processor of claim 67, wherein the first and second ALU share the source operand stored in the first input register of the first ALU substantially simultaneously.

84-91. (CANCELLED)

92. (PREVIOUSLY PRESENTED) A method for use with a processor, the method comprising:

providing a multiplexer (MUX) having a first and second MUX input and a MUX output;

coupling the first MUX input to a first input of a first ALU, the first ALU capable of addressing a memory to retrieve a source operand from the memory;

coupling a second MUX input to a first input register of a second ALU, the second ALU not capable of addressing the memory; and

directly providing, by the MUX output, a first input to the second ALU, the MUX permitting both the first and second ALU to share the source operand retrieved from the memory by the first ALU.

93. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:

defining a register decode value within an instruction set that specifies source operand bypassing, such that the MUX, in response to the register decode value that specifies source operand bypassing, selects the first MUX input coupled to the first input of the first ALU as the MUX output, the MUX output providing the first input to the second ALU.

94. (PREVIOUSLY PRESENTED) The method of claim 93, wherein the source operand bypassing value allows the second execution unit to receive data stored at an effective memory address specified by a displacement operand in the previous instruction executed by the first execution unit.

95. (CURRENTLY AMENDED) The method of claim 92, further comprising:

communicating with ~~a~~ the memory;

storing intermediate operands in a register file; and

identifying an instruction decode stage for coupling the register file to the first and second input registers of the first and second ALUs to provide intermediate operands as source operands, and for coupling a memory bus to the first input register of the of the first ALU to provide source operands from the memory.

96. (PREVIOUSLY PRESENTED) The method of claim 95, further comprising:

providing, by the first input register of the first ALU, source operands from the memory to both the first input to the first ALU and to the first MUX input, thereby permitting the first input to the second ALU to share the source operands from the memory directly from the first input register of the first ALU.

97. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:

including a pipeline of the processor, the pipeline having a plurality of stages including instruction decode, writeback, and execution stages, the execution stage having a first and second execution unit, each having one of the first and second ALUs, respectively.

98. (PREVIOUSLY PRESENTED) The method of claim 97, further comprising:

defining a register decode value that defines result bypassing of a result from a previous instruction executing in pipeline stages of the processor.

99. (PREVIOUSLY PRESENTED) The method of claim 98, further comprising:

identifying a pipeline stage register for use as a source operand in an instruction containing the register decode value by directly addressing a result register.

100. (PREVIOUSLY PRESENTED) The method of claim 99, further comprising:

explicitly controlling data flow within the pipeline stages of the processor through the use of a register result bypass (RRB) operand in the register decode value.

101. (PREVIOUSLY PRESENTED) The method of claim 100, wherein the step of explicitly controlling comprises:

retrieving data from the first execution unit; and

returning the data to an input of the first and second execution units as specified by the RRB operand, thereby bypassing write-back of the data to either a register file or memory at the writeback stage.

102. (PREVIOUSLY PRESENTED) The method of claim 101, wherein the step of identifying further comprises:

explicitly specifying the pipeline stage register to be used as the source operand for the instruction.

103. (PREVIOUSLY PRESENTED) The method of claim 102, further comprising:

encoding the RRB operand in fewer bits than a regular register operand.

104. (PREVIOUSLY PRESENTED) The method of claim 97, further comprising:

defining a register decode value that defines source operand bypassing of source operand data.

105. (PREVIOUSLY PRESENTED) The method of claim 104, further comprising:

identifying a pipeline stage register for use as a source operand in an instruction containing the register decode value by directly addressing a source register.

106. (PREVIOUSLY PRESENTED) The method of claim 105, further comprising:

sharing source operand data among the first and second execution units of the pipelined processor through the use of a source bypass (RISB) operand in the register decode value.

107. (PREVIOUSLY PRESENTED) The method of claim 106, wherein the step of sharing further comprises:

receiving data at the second execution unit, the data stored at a memory address specified by a displacement operand in a previous instruction executed by the first execution unit of the processor.

108. (PREVIOUSLY PRESENTED) The method of claim 107, wherein the step of sharing further comprises:

realizing two memory references through the use of a single bus operation over a local bus.

109. (PREVIOUSLY PRESENTED) The method of claim 108, wherein the step of sharing further comprises:

encoding the RISB operand with substantially fewer bits than those needed for a displacement address.

110. (PREVIOUSLY PRESENTED) The method of claim 92, further comprising:

sharing a source operand stored in a first input register of the first ALU at the first and second ALU substantially simultaneously.

111. (CURRENTLY AMENDED) An apparatus, comprising:

~~means for providing~~ a multiplexer (MUX) having a first and second MUX input and a MUX output;

means for coupling the first MUX input to a first input for a first ALU, the first ALU capable of addressing a memory to retrieve a source operand from the memory;

means for coupling a second MUX input to a first input register for a second ALU, the second ALU not capable of addressing the memory; and

means for directly providing~~, by the MUX output,~~ a first input to the second ALU, the ~~MUX~~ means for directly providing permitting both the first and second ALU to share the source operand retrieved from the memory by the first ALU.

112. (PREVIOUSLY PRESENTED) The apparatus of claim 111, further comprising:

means for defining a register decode value within an instruction set that specifies source operand bypassing, such that the MUX, in response to the register decode value that specifies source operand bypassing, selects the first MUX input coupled to the first input of the first ALU as the MUX output, the MUX output providing the first input to the second ALU.

113. (PREVIOUSLY PRESENTED) The apparatus of claim 112, wherein the source operand bypassing value allows the second execution unit to receive data stored at an effective memory address specified by a displacement operand in the previous instruction executed by the first execution unit.

114. (CURRENTLY AMENDED) The apparatus of claim 111, further comprising:

means for communicating with ~~a~~ the memory;

means for storing intermediate operands; and

means for identifying an instruction decode stage for coupling the register file to the first and second input registers of the first and second ALUs to provide intermediate operands as source operands, and for coupling a memory bus to the first input register of the of the first ALU to provide source operands from the memory.

115. (CURRENTLY AMENDED) The apparatus of claim 114, further comprising:

means for providing~~, by the first input register of the first ALU,~~ source operands from the memory to both the first input to the first ALU and to the first MUX input, thereby permitting the first input to the second ALU to share the source operands from the memory directly from the first input register of the first ALU.

116. (PREVIOUSLY PRESENTED) The apparatus of claim 111, further comprising:

means for sharing a source operand stored in a first input register of the first ALU at the first and second ALU substantially simultaneously.

117. (CANCELLED)

118. (PREVIOUSLY PRESENTED) A processor, comprising:

a first ALU, the first ALU capable of addressing a memory to retrieve a source operand from the memory;

a second ALU, the second ALU not capable of addressing the memory; and

a circuit to couple a first input of the first ALU to a first input of the second ALU to provide an operand retrieved from the memory by the first ALU as an input to the second ALU.

119. (PREVIOUSLY PRESENTED) The processor of claim 118, wherein the circuit further comprises:

a multiplexer, the multiplexer having an input connected to the first input of the first ALU and an output connected to the first input of the second ALU.

120. (PREVIOUSLY PRESENTED) A processor, comprising:

a first ALU, the first ALU capable of addressing a memory to retrieve an operand

a second ALU, the second ALU not capable of addressing the memory;

a first circuit capable of providing an operand retrieved by the first ALU from a memory as an input to the second ALU;

a second circuit capable of providing a result from either the first ALU or the second ALU as an input to the second ALU; and

an instruction set having a register decode value which is capable of selecting as an input to the second ALU either the operand or the result.

121. (PREVIOUSLY PRESENTED) The processor of claim 120, wherein the circuit further comprises:

a multiplexer, the multiplexer having a first input connected to the first input of the first ALU, a second input coupled to the result, and an output connected to the input of the second ALU, the multiplexer selecting whether the second ALU receives the operand or the result as an input.

122. (PREVIOUSLY PRESENTED) The processor of claim 120, further comprising:

a circuit providing the result to both the first ALU and the second ALU.

123. (PREVIOUSLY PRESENTED) A method for operating a processor, comprising:

addressing a memory by a first ALU to retrieve a source operand from the memory;

providing a second ALU, the second ALU not capable of addressing the memory; and

coupling a first input of the first ALU to a first input of the second ALU to provide an operand retrieved from the memory by the first ALU as an input to the second ALU.

124. (PREVIOUSLY PRESENTED) The method of claim 123, further comprising:

connecting an input of a multiplexer to the first input of the first ALU; and

connecting an output of the multiplexer to the first input of the second ALU.

125. (PREVIOUSLY PRESENTED) A method for operating a processor, comprising:

providing an operand retrieved by a first ALU from a memory as an input to a second ALU, the second ALU not capable of addressing the memory;

providing a result from either the first ALU or the second ALU as an input to the second ALU; and

selecting, by an instruction set having a register decode value, either the operand or the result as an input to the second ALU.

126. (PREVIOUSLY PRESENTED) The method of claim 125, further comprising:

connecting a first input of a multiplexer to the first input of the first ALU;

connecting a second input of the multiplexer to the result; and

connecting an output of the multiplexer to the input of the second ALU, the multiplexer selecting whether the second ALU receives the operand or the result as an input.

127. (PREVIOUSLY PRESENTED) The method of claim 125, further comprising:

providing the result to both the first ALU and the second ALU.